# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-047998
(43)Date of publication of application : 18.02.2000

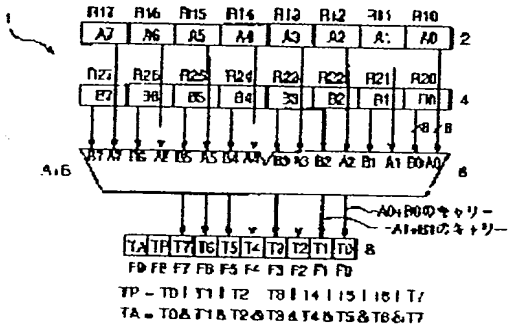
(51)Int.Cl. G06F 15/16
G06F 9/38
G06F 15/80

(21)Application number : 10-217027
(22)Date of filing : 31.07.1998

(71)Applicant : RICOH CO LTD
(72)Inventor : OTEGI SUGITAKA
HARA KAZUHIKO
YAMAURA SHINICHI
KADOWAKI YUKIO

## (54) SIMD SYSTEM ARITHMETIC UNIT AND ARITHMETIC PROCESSOR

(57)Abstract:
PROBLEM TO BE SOLVED: To make a succeeding processing selectively executable by executing calculation through the use of two kinds of data to be stored in respective corresponding data storing parts between two input means and storing condition flags corresponding to the arithmetic results in the corresponding flag storing parts of an output means.
SOLUTION: The arithmetic unit 1 simultaneously executes calculation being common to respective data classes by an arithmetic part 6 through the use of input data A0-A7 stored in the respective data storing parts R10-R17 of a first input register 2 and input data B0-B7 which are stored in the respective data storing parts R20-R27 of the second input register 4 and stores the flags T0-T7 corresponding to the arithmetic results in the flag storing parts F0-F7 of an output register 8. The flags TP and TA corresponding to the result of OR operation in the output flags T0-T7 which are stored in the flag storing parts R0-R7 are stored in the flag storing parts F8 and F9 of the output register 8.



LEGAL STATUS

| | |
|---|---|
| [Date of request for examination] | 20.05.2003 |
| [Date of sending the examiner's decision of rejection] | |
| [Kind of final disposal of application other than the examiner's decision of rejection or application converted registration] | |
| [Date of final disposal for application] | |
| [Patent number] | 3652518 |
| [Date of registration] | 04.03.2005 |
| [Number of appeal against examiner's decision of rejection] | |
| [Date of requesting appeal against examiner's decision of rejection] | |
| [Date of extinction of right] | |

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号
特開2000－47998
（P2000－47998A）
(43)公開日　平成12年２月18日(2000.2.18)

| (51)Int.Cl.[7] | 識別記号 | ＦＩ | | テーマコード（参考） |
|---|---|---|---|---|
| Ｇ０６Ｆ 15/16 | ６１０ | Ｃ０６Ｆ 15/16 | ６１０Ａ | |
| 9/38 | ３７０ | 9/38 | ３７０Ｘ | |
| 15/80 | | 15/80 | | |

審査請求　未請求　請求項の数17　ＯＬ　（全 14 頁）

(21)出願番号　特願平10－217027

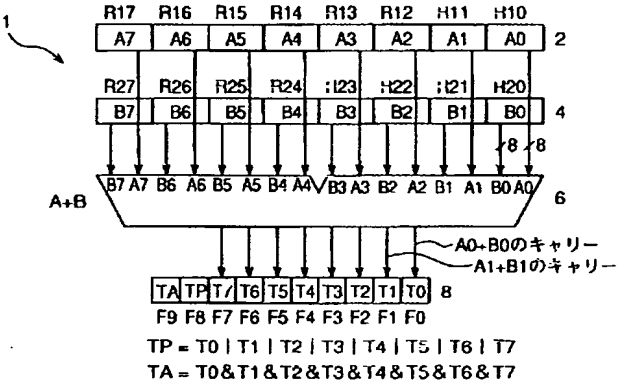
(22)出願日　平成10年７月31日(1998.7.31)

(71)出願人　000006747
株式会社リコー
東京都大田区中馬込１丁目３番６号
(72)発明者　樽木　杉高
東京都大田区中馬込１丁目３番６号　株式会社リコー内
(72)発明者　原　和彦
東京都大田区中馬込１丁目３番６号　株式会社リコー内
(74)代理人　100062144
弁理士　青山　葆　（外１名）

最終頁に続く

(54)【発明の名称】　ＳＩＭＤ方式の演算器及び演算処理装置

(57)【要約】
【課題】　ＳＩＭＤ方式の演算器において、演算単位ごとに異なる処理を行う。また、回路規模の有効利用を目指す。
【解決手段】　ＳＩＭＤ型演算器において、各演算単位の演算結果に対応するフラグを出力する。そのフラグを、① 演算対象データとする、② 条件分岐処理の判断条件とする、③ 後続の演算の制御条件とすることにより、演算単位ごとに異なる処理を行う。また、上記フラグの論理和フラグ論理積フラグを求め、条件分岐処理等の後続処理にて利用する。ブロードキャスト方式の導入により、処理命令の簡素化、処理速度の向上を目指す。さらに、入出力手段を分割して用いて、それぞれにデータを格納し演算に繋げる、あるいは演算後それぞれにデータを格納することにより、回路規模の有効利用が実現できる。



【特許請求の範囲】
【請求項１】　２つの入力手段と１つの出力手段をもつ演算器であって、
第１の入力手段及び第２の入力手段はいずれも所定のビット長の長さであり、かつ格納するデータのビット長に応じて個数及びビット長が変化するデータ格納部を有し、
出力手段は一方の入力手段における上記データ格納部の個数以上の個数まで長さ１ビットであるフラグ格納部を有し、
第１の入力手段の各データ格納部に格納されるデータとこれに対応する第２の入力手段の各データ格納部に格納されるデータとを用いて同時に各データの組に共通の演算を行う演算器において、
２つの入力手段間で対応する各データ格納部に格納される２つのデータを用いてそれぞれ演算した結果に対応する条件フラグを出力手段の対応するフラグ格納部に格納する演算器。
【請求項２】　請求項１に記載の演算器において、
第２の入力手段は少なくとも第１の入力手段のデータ格納部のビット長以上の長さであり、かつ第１の入力手段のデータ格納部と長さが等しい１個のデータ格納部を有し、
第１の入力手段の各データ格納部に格納されるデータと第２の入力手段の１個のデータ格納部に格納されるデータとを用いて同時に各データの組に共通の演算を行い、
第１の入力手段の各データと第２の入力手段のデータとを用いて演算した結果に対応する条件フラグを出力手段の対応するフラグ格納部に格納する演算器。
【請求項３】　２つの入力手段と１つの出力手段をもつ演算器であって、
第１の入力手段は所定のビット長の長さであり、かつ格納するデータのビット長に応じて個数及びビット長が変化するデータ格納部を有し、
第２の入力手段は第１の入力手段における上記データ格納部の個数以上の個数まで長さ１ビットであるフラグ格納部を有し、かつそれぞれのフラグ格納部中に先行する演算における演算結果に対応した条件フラグを格納し、
第１の入力手段の各データ格納部に格納されるデータとこれに対応する第２の入力手段の各フラグ格納部に格納される条件フラグとを用いて同時に各データと条件フラグの組に共通の演算を行い出力手段に格納する演算器。
【請求項４】　少なくとも１つの入力手段と、１つの出力手段をもつ演算器であって、
入力手段及び出力手段は所定のビット長の長さであり、かつ格納するデータのビット長に応じて個数及びビット長が変化するデータ格納部を有し、
入力手段の各データ格納部に格納されるデータを用いて同時に各データに共通の演算を行った結果得られたデータを対応する出力手段のデータ格納部に格納する演算器において、
演算制御手段が入力手段における上記データ格納部の個数以上の個数まで長さ１ビットであるフラグ格納部を有し、かつそれぞれのフラグ格納部中に先行する演算における演算結果に対応した条件フラグを格納し、
各フラグ格納部が入力手段の各データ格納部に対応しており、入力手段の各データ格納部に格納されるデータが演算に用いられる際に、当該データ格納部に対応する各フラグ格納部に格納される条件フラグの内容によりデータ毎に演算に条件が与えられる演算器。
【請求項５】　請求項１または請求項２に記載の演算器において、すべての条件フラグの論理和を求めて条件論理和フラグとし、それを出力手段の対応するフラグ格納部に格納する演算器。
【請求項６】　請求項１または請求項２に記載の演算器において、すべての条件フラグの論理積を求めて条件論理積フラグとし、それを出力手段の対応するフラグ格納部に格納する演算器。
【請求項７】　請求項１に記載の演算器を備えた中央演算処理装置。
【請求項８】　請求項２に記載の演算器を備えた中央演算処理装置。
【請求項９】　請求項１に記載の演算器と請求項３に記載の演算器とを備えた中央演算処理装置。
【請求項１０】　請求項１に記載の演算器と請求項４に記載の演算器とを備えた中央演算処理装置。
【請求項１１】　請求項１に記載の演算器と請求項５に記載の演算器と請求項６に記載の演算器とを備えた中央演算処理装置。
【請求項１２】　請求項１に記載の演算器と請求項２に記載の演算器と請求項３に記載の演算器と請求項４に記載の演算器と請求項５に記載の演算器と請求項６に記載の演算器とを備えた中央演算処理装置。
【請求項１３】　生成された複数の上記条件フラグを分岐処理の判断条件とする条件分岐処理機能を備える、請求項７、請求項８、請求項９、請求項１０、請求項１１、または請求項１２に記載の中央演算処理装置。
【請求項１４】　請求項１または請求項２に記載の出力手段上で最上位（あるいは最下位）に位置する“１”を格納したフラグ格納部の位置を数値化する機能を備える、請求項７、請求項８、請求項９、請求項１０、請求項１１、請求項１２または請求項１３に記載の中央演算処理装置。
【請求項１５】　請求項１または請求項２に記載の出力手段上で最上位（あるいは最下位）に位置する“０”を格納したフラグ格納部の位置を数値化する機能を備える、請求項７、請求項８、請求項９、請求項１０、請求項１１、請求項１２または請求項１３に記載の中央演算処理装置。
【請求項１６】　請求項５に記載の条件論理和フラグを

分岐処理の判断条件とする条件分岐処理機能を備える、請求項１１または請求項１２に記載の中央演算処理装置。

【請求項１７】　請求項５に記載の条件論理積フラグを分岐処理の判断条件とする条件分岐処理機能を備える、請求項１１または請求項１２に記載の中央演算処理装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、ＳＩＭＤ（Ｓｉｎｇｌｅ　Ｉｎｓｔｒｕｃｔｉｏｎ　Ｍｕｌｔｉｐｌｅ　Ｄａｔａ）方式を用いた演算器及びこの演算器を備えた演算処理装置（以下、ＣＰＵという。）に関する。

【０００２】

【従来の技術】ＣＰＵにおいて複数のデータを並列処理する方式としてＳＩＭＤ方式がある。ＳＩＭＤ方式では、ＣＰＵ中の演算器において、１つの演算命令によって複数の演算が並列に制御される。また、命令供給装置や命令制御装置の共有化や、処理実行時間短縮が図れるという利点がある。

【０００３】

【発明が解決しようとする課題】他方、ＳＩＭＤ方式の演算器においては、演算単位により、演算対象データは異なるが、それら複数の演算の処理機能は同じである。つまり演算単位ごとに異なる処理をすることができない。例えば、あるデータ群に対してあるデータと比較演算した結果から判断して、一致した演算のデータだけ“０”に置き換えるといったことが、困難である。

【０００４】また、ＳＩＭＤ方式では、処理に際して、１演算単位に１演算器を割り当て全体で演算器を複数用いることが多いが、このことが、演算データのサイズによっては不合理に大きな回路規模を必要とすることがある。例えば、１６ビットデータの演算が多く、まれに６４ビットデータの演算処理が必要になるような場合でも、ＣＰＵにおいては最大データ幅の演算器を最大並列数まで備えておかねばならず、回路規模、装置規模が有効に使われないことがある。

【０００５】本発明は、演算単位の演算結果に対応する条件フラグによって後続の処理を選択的に実行するＳＩＭＤ方式の演算器及び演算処理装置を提供することを目的とする。

【０００６】本発明はまた、演算データ単位に後続の処理を選択的に実行するＳＩＭＤ方式の演算器及び演算処理装置において、ビット長の短いデータを多く並列演算処理するとしても、最大データ幅を扱える演算器をデータの最大並列処理数まで必ずしも設けることなく、有効に利用できる回路規模を持つことを可能とすることを別の目的とする。

【０００７】

【課題を解決するための手段】本発明の第一の形態は、２つの入力手段と１つの出力手段をもつＳＩＭＤ型演算器であって、そこでは第１の入力手段及び第２の入力手段はいずれも所定のビット長の長さであり、かつ格納するデータのビット長に応じて個数及びビット長が変化するデータ格納部を有し、出力手段は一方の入力手段における上記データ格納部の個数以上の個数まで長さ１ビットであるフラグ格納部を有する。このＳＩＭＤ型演算器では、第１の入力手段の各データ格納部に格納されるデータとこれに対応する第２の入力手段の各データ格納部に格納されるデータとを用いて同時に各データの組に共通の演算を行うのであるが、本発明のＳＩＭＤ型演算器では、２つの入力手段間で対応する各データ格納部に格納される２つのデータを用いてそれぞれ演算しその演算結果に対応する条件フラグを出力手段の対応するフラグ格納部に格納する。

【０００８】上記第一の形態の演算器において、第２の入力手段が少なくとも第１の入力手段のデータ格納部のビット長以上の長さであり、かつ第１の入力手段のデータ格納部と長さが等しい１個のデータ格納部を有し、第１の入力手段の各データ格納部に格納されるデータと第２の入力手段の１個のデータ格納部に格納されるデータとを用いて同時に各データの組に共通の演算を行うものであってもよい。

【０００９】上記第一の形態の演算器の出力手段上にて、すべての条件フラグの論理和を求めて条件論理和フラグとし、そのフラグを出力手段上の対応するフラグ格納部に格納することもできる。同様に、すべての条件フラグの論理積を求めて条件論理積フラグとし、そのフラグを出力手段上の対応するフラグ格納部に格納することもできる。

【００１０】本発明の第二の形態は、２つの入力手段と１つの出力手段をもつＳＩＭＤ型演算器であって、そこでは、第１の入力手段は上記第一の形態の演算器の場合のそれと同じ構成であり、第２の入力手段では上記第一の形態の演算器での出力手段を用いる。よって、第１の入力手段の各データ格納部に格納されるデータとこれに対応する第２の入力手段の各フラグ格納部に格納される条件フラグとを用いて同時に各データと条件フラグの組に共通の演算を行い、その演算結果を出力手段に格納する。

【００１１】本発明の第三の形態は、少なくとも１つの入力手段と、１つの出力手段をもつＳＩＭＤ型演算器であって、そこでは、入力手段及び出力手段は所定のビット長の長さであり、かつ格納するデータのビット長に応じて個数及びビット長が変化するデータ格納部を有する。このＳＩＭＤ型演算器は、入力手段の各データ格納部に格納されるデータを用いて同時に各データに共通の演算を行った結果得られたデータを対応する出力手段のデータ格納部に格納するのであるが、本発明のＳＩＭＤ型演算器では、上記第一の形態の演算器で出力される条

件フラグ群を格納する出力手段の各フラグ格納部が当該第三の形態の入力手段上の各データ格納部に対応しており、入力手段の各データ格納部に格納されるデータが演算される際に、当該データ格納部に対応する各フラグ格納部に格納される条件フラグの内容によりデータ毎に演算に条件が与えられる。

【００１２】本発明の第四の形態は、上記第一の形態の演算器、第二の形態の演算器、第三の形態の演算器を備えたＣＰＵである。

【００１３】本発明の第五の形態は、上記第一の形態の演算器にて生成された複数の上記条件フラグを分岐処理の判断条件とする条件分岐処理機能を備えたＣＰＵである。

【００１４】本発明の第六の形態は、上記第一の形態の演算器の出力手段上で最上位（あるいは最下位）に位置する“１”（あるいは“０”）を格納したフラグ格納部の位置を数値化する機能を備えたＣＰＵである。

【００１５】本発明の第七の形態は、上記第一の形態の演算器にて出力手段上に格納される条件論理和フラグを分岐処理の判断条件とする条件分岐処理機能を備えたＣＰＵである。同様に、条件論理積フラグを分岐処理の判断条件とする条件分岐処理機能を備えることもできる。

【００１６】

【発明の実施の形態】以下、添付図面を参照して本発明の実施の形態を説明する。本発明の第１の実施の形態のＳＩＭＤ型演算器（以下、演算器という。）を図１に示す。演算器１は、第１の入力レジスタ２と、第２の入力レジスタ４と、演算部６と、出力レジスタ８を有する。２つの入力レジスタ２、４のビット長は６４ビットである。

【００１７】図１の演算器１では、第１と第２の入力レジスタ２、４はいずれも８ビットのビット長を有する８個のデータ格納部Ｒ１０～Ｒ１７、Ｒ２０～Ｒ２７で構成されており、各データ格納部に所定の演算データＡ０～Ａ７、Ｂ０～Ｂ７が格納できるようにしてある。出力レジスタ８は、１ビットのビット長を有する１０個のフラグ格納部Ｆ０～Ｆ９を有し、各フラグ格納部にそれぞれ所定のフラグ（Ｔ０～Ｔ７、ＴＰ、ＴＡ）が格納できるようにしてある。

【００１８】この演算器１では、第１の入力レジスタ２の各データ格納部Ｒ１０～Ｒ１７に格納された入力データＡ０～Ａ７と第２の入力レジスタ４の各データ格納部Ｒ２０～Ｒ２７に格納された入力データＢ０～Ｂ７とをそれぞれ用いて演算部６において同時に各データの組に共通の演算が行われ、その演算結果に対応したフラグＴ０～Ｔ７（０又は１）が出力レジスタ８のフラグ格納部Ｆ０～Ｆ７に格納される。出力レジスタ８のフラグ格納部Ｆ８には、フラグ格納部Ｆ０～Ｆ７に格納されている出力フラグＴ０～Ｔ７の論理和演算の結果に対応したフラグＴＰ（０又は１）が格納される。他方、出力レジスタのデータ格納部Ｆ９には、フラグ格納部Ｆ０～Ｆ７に格納されている出力フラグＴ０～Ｔ７の論理積演算の結果に対応したフラグＴＡ（０又は１）が格納される。

【００１９】出力レジスタ８のフラグ格納部Ｆ０～Ｆ７に格納されるフラグＴ０～Ｔ７について具体的に説明する。例えば、演算部６で２つの入力データＡ０、Ｂ０を加算する場合、これらのデータを加算して得られた結果が８ビットを超えると（すなわち、桁上がりを生じると）、対応する出力フラグ格納部Ｆ０にフラグ１を格納する。逆に、加算結果が８ビット以下の場合（即ち、桁上がりを生じない場合）、対応する出力データ格納部Ｆ０にフラグ０を格納する。

【００２０】図２は図１に示す演算器の変形例を示す。第１と第２の入力レジスタ１２、１４はいずれも１６ビットのビット長を有する４個のデータ格納部Ｒ１０～Ｒ１３、Ｒ２０～Ｒ２３で構成されており、各データ格納部に所定の演算データＡ０～Ａ３、Ｂ０～Ｂ３が格納できるようにしてある。出力レジスタ１８は、１ビットのビット長を有する６個のフラグ格納部Ｆ０～Ｆ５を有し、各フラグ格納部にそれぞれ所定のフラグ（Ｔ０～Ｔ３、ＴＰ、ＴＡ）が格納できるようにしてある。ここで、出力レジスタ１８のフラグ格納部Ｆ０～Ｆ３に格納されるフラグＴ０～Ｔ３は、上記図１の実施の形態の演算器１のフラグＴ０～Ｔ７と同様に求められ格納される。出力レジスタ１８のフラグ格納部Ｆ４にはフラグＴＰが格納されるが、上記図１の実施の形態と概略同様で、フラグ格納部Ｆ０～Ｆ３に格納されている出力フラグＴ０～Ｔ３の論理和演算の結果に対応したものである。同様に、出力レジスタ１８のフラグ格納部Ｆ５にはフラグＴＡが格納され、フラグ格納部Ｆ０～Ｆ３に格納されている出力フラグＴ０～Ｔ３の論理積演算の結果に対応する。

【００２１】図３も図１に示す演算器の変形例を示す。第１と第２の入力レジスタ２２、２４はいずれも３２ビットのビット長を有する２個のデータ格納部Ｒ１０～Ｒ１１、Ｒ２０～Ｒ２１で構成されており、各データ格納部に所定の演算データＡ０～Ａ１、Ｂ０～Ｂ１が格納できるようにしてある。出力レジスタ２８は、１ビットのビット長を有する４個のフラグ格納部Ｆ０～Ｆ３を有し、各フラグ格納部にそれぞれ所定のフラグ（Ｔ０～Ｔ１、ＴＰ、ＴＡ）が格納できるようにしてある。ここで、出力レジスタ２８のフラグ格納部Ｆ０～Ｆ１に格納されるフラグＴ０～Ｔ１は、上記図１の実施の形態の演算器１のフラグＴ０～Ｔ７と同様に求められ格納される。出力レジスタ２８のフラグ格納部Ｆ２にはフラグＴＰが格納されるが、上記図１の実施の形態と概略同様で、フラグ格納部Ｆ０～Ｆ１に格納されている出力フラグＴ０～Ｔ１の論理和演算の結果に対応したものである。同様に、出力レジスタ２８のフラグ格納部Ｆ３にはフラグＴＡが格納され、フラグ格納部Ｆ０～Ｆ１に格納

されている出力フラグＴ０～Ｔ１の論理積演算の結果に対応する。

【００２２】図４もまた、図１に示す演算器の変形例を示す。第１と第２の入力レジスタ３２、３４はいずれも６４ビットのビット長を有する１個のデータ格納部Ｒ１０、Ｒ２０で構成されており、各データ格納部に所定の演算データＡ０、Ｂ０が格納できるようにしてある。出力レジスタ３８は、１ビットのビット長を有する１個のフラグ格納部Ｆ０を有し、そのフラグ格納部には所定のフラグＴ０が格納できるようにしてある。ここで、出力レジスタ３８のフラグ格納部Ｆ０に格納されるフラグＴ０は、上記図１の実施の形態の演算器１のフラグＴ０～Ｔ７と同様に求められ格納される。

【００２３】これらの演算器によって上記のようにして出力された演算結果に対応するフラグ（以下、条件フラグという。）を用いると、後続処理において演算単位ごとに異なる処理を行うことができる。また、当該条件フラグによる条件分岐処理が可能になる。

【００２４】また、これらの演算器によって上記のようにして出力された条件フラグの論理和演算の結果に対応したフラグ（以下条件論理和フラグという。）ＴＰを用いると、当該条件論理和フラグによる条件分岐処理が可能になる。同様に、条件フラグの論理積演算の結果に対応したフラグ（以下条件論理積フラグという。）ＴＡを用いると、当該条件論理積フラグによる条件分岐処理が可能になる。

【００２５】図１の形態では、６４ビットのビット長を有する入力レジスタを備えた演算器を１個だけ用意し、その１個の演算器の内部で８つの演算を並列的に行うことができるので、同一ビット長の入力レジスタを有する演算器を８個も用意する必要がない。その結果、小さい回路規模を実現できる。図１の形態のみならず、図２及び図３においても同様である。

【００２６】図１、図２、図３及び図４では、演算の結果に対応するフラグとして、桁上がりを示すキャリーの場合を示したが、この他に、演算結果があふれた場合に対応するオーバーフローフラグ、演算結果が‘０’のときに対応するゼロフラグ、演算結果が負のときに対応するネガティブフラグ等でもよい。

【００２７】本発明の第２の実施の形態の演算器を図５に示す。演算器４０は、第１の入力レジスタ４２と、第２の入力レジスタ４４と、演算部４６と、出力レジスタ４８とを有するが、ここで、第２の入力レジスタ４４は上記第１の実施の形態での演算器１における出力レジスタ（以下、フラグレジスタという。）８であり、フラグレジスタ８に格納される条件フラグＴ０～Ｔ７を入力データ、つまり演算対象データとする。第１の入力レジスタ４２、出力レジスタ４８のビット長は６４ビットである。

【００２８】図５の演算器４０では、第１の入力レジスタ４２は、８ビットのビット長を有する８個のデータ格納部Ｒ１０～Ｒ１７で構成されており、各データ格納部に所定の演算データＡ０～Ａ７が格納できるようにしてある。第２の入力レジスタ４４、即ちフラグレジスタ８は、１ビットのビット長を有する少なくとも８個のフラグ格納部Ｆ０～Ｆ７で構成されており、各フラグ格納部に上記第１の形態での演算器１における出力データたる条件フラグＴ０～Ｔ７を格納している。出力レジスタ４８は、８ビットのビット長を有する８個のデータ格納部Ｒ３０～Ｒ３７で構成されており、各データ格納部に所定の演算結果データＺ０～Ｚ７が格納できるようにしてある。

【００２９】この演算器４０では、第１の入力レジスタ４２の各データ格納部Ｒ１０～Ｒ１７に格納された入力データＡ０～Ａ７と第２の入力レジスタ４４（即ちフラグレジスタ８）の各フラグ格納部Ｆ０～Ｆ７に格納された条件フラグＴ０～Ｔ７とをそれぞれ用いて演算部４６において同時に各データと条件フラグの組に共通の演算が行われ、その演算結果Ｚ０～Ｚ７が出力レジスタ４８のデータ格納部Ｒ３０～Ｒ３７に格納される。例えば、演算部４６で２つの入力データＡ０、Ｔ０を加算する場合、これらのデータを加算して得られた結果Ｚ０が出力データ格納部Ｒ３０に格納される。

【００３０】図６は図５に示す演算器の変形例を示す。第１の入力レジスタ５２は、１６ビットのビット長を有する４個のデータ格納部Ｒ１０～Ｒ１３で構成されており、各データ格納部に所定の演算データＡ０～Ａ３が格納できるようにしてある。第２の入力レジスタ５４、即ちフラグレジスタ１８は、１ビットのビット長を有する少なくとも４個のフラグ格納部Ｆ０～Ｆ３で構成されており、各フラグ格納部に上記第１の形態での演算器１０における出力データたる条件フラグＴ０～Ｔ３を格納している。出力レジスタ５８は、１６ビットのビット長を有する４個のデータ格納部Ｒ３０～Ｒ３３で構成されており、各データ格納部に所定の演算結果データＺ０～Ｚ３が格納できるようにしてある。

【００３１】この演算器５０では、第１の入力レジスタ５２の各データ格納部Ｒ１０～Ｒ１３に格納された入力データＡ０～Ａ３と第２の入力レジスタ５４（即ちフラグレジスタ１８）の各フラグ格納部Ｆ０～Ｆ３に格納された条件フラグＴ０～Ｔ３とをそれぞれ用いて演算部５６において同時に各データと条件フラグの組に共通の演算が行われ、その演算結果Ｚ０～Ｚ３が出力レジスタ５８のデータ格納部Ｒ３０～Ｒ３３に格納される。例えば、演算部５６で２つの入力データＡ０、Ｔ０を加算する場合、これらのデータを加算して得られた結果Ｚ０が出力データ格納部Ｒ３０に格納される。

【００３２】図７も図５に示す演算器の変形例を示す。第１の入力レジスタ６２は、３２ビットのビット長を有する２個のデータ格納部Ｒ１０～Ｒ１１で構成されてお

り、各データ格納部に所定の演算データＡ０～Ａ１が格納できるようにしてある。第２の入力レジスタ６４、即ちフラグレジスタ２８は、１ビットのビット長を有する少なくとも２個のフラグ格納部Ｆ０～Ｆ１で構成されており、各フラグ格納部に上記第１の形態での演算器２０における出力データたる条件フラグＴ０～Ｔ１を格納している。出力レジスタ６８は、３２ビットのビット長を有する２個のデータ格納部Ｒ３０～Ｒ３１で構成されており、各データ格納部に所定の演算結果データＺ０～Ｚ１が格納できるようにしてある。

【００３３】この演算器６０では、第１の入力レジスタ６２の各データ格納部Ｒ１０～Ｒ１１に格納された入力データＡ０～Ａ１と第２の入力レジスタ６４（即ちフラグレジスタ２８）の各フラグ格納部Ｆ０～Ｆ１に格納された条件フラグＴ０～Ｔ１とをそれぞれ用いて演算部６６において同時に各データと条件フラグの組に共通の演算が行われ、その演算結果Ｚ０～Ｚ１が出力レジスタ６８のデータ格納部Ｒ３０～Ｒ３１に格納される。例えば、演算部６６で２つの入力データＡ０、Ｔ０を加算する場合、これらのデータを加算して得られた結果Ｚ０が出力データ格納部Ｒ３０に格納される。

【００３４】図８もまた図５に示す演算器の変形例を示す。第１の入力レジスタ７２は、６４ビットのビット長を有する１個のデータ格納部Ｒ１０で構成されており、所定の演算データＡ０が格納できるようにしてある。第２の入力レジスタ７４、即ちフラグレジスタ３８は、１ビットのビット長を有する少なくとも１個のフラグ格納部Ｆ０で構成されており、上記第１の形態での演算器３０における出力データたる条件フラグＴ０を格納している。出力レジスタ７８は、６４ビットのビット長を有する１個のデータ格納部Ｒ３０で構成されており、所定の演算結果データＺ０が格納できるようにしてある。

【００３５】この演算器７０では、第１の入力レジスタ７２のデータ格納部Ｒ１０に格納された入力データＡ０と第２の入力レジスタ７４（即ちフラグレジスタ）のフラグ格納部Ｆ０に格納された条件フラグＴ０とを用いて演算部７６において演算が行われ、その演算結果Ｚ０が出力レジスタ７８のデータ格納部Ｒ３０に格納される。

【００３６】このように構成された演算器によれば、先行して実行したＳＩＭＤ型演算の結果を後続するＳＩＭＤ型演算に演算単位で反映させるのが容易になる。

【００３７】図５の形態では、６４ビットのビット長を有する入力レジスタを備えた演算器を１個だけ用意し、その１個の演算器の内部で８つの演算を並列的に行うことができるので、同一ビット長の入力レジスタを有する演算器を８個も用意する必要がない。その結果、小さい回路規模を実現できる。図５の形態のみならず、図６及び図７においても同様である。

【００３８】本発明の第３の実施の形態の演算器を図９に示す。演算器８０は、第１の入力レジスタ８２と、第２の入力レジスタ８４と、演算部８６と、出力レジスタ８８とを有する。本構成は、本発明の第２の実施の形態に、第１の実施の形態を組み合わせたものとなっている。つまり、第２の入力レジスタ８４は、第２の実施の形態と同一で、上記第１の実施の形態での演算器１におけるフラグレジスタ８であるが、出力レジスタ８８は、第１の実施の形態同様、演算結果に対応する条件フラグと、条件フラグの内容により決定される条件論理和フラグ及び条件論理積フラグとを格納する。

【００３９】図９の演算器８０では、第１の入力レジスタ８２は、８ビットのビット長を有する８個のデータ格納部Ｒ１１～Ｒ１７で構成されており、各データ格納部に所定の演算データＡ０～Ａ７が格納できるようにしてある。第２の入力レジスタ８４、即ちフラグレジスタ８は、１ビットのビット長を有する少なくとも８個のフラグ格納部Ｆ０～Ｆ７で構成されており、各フラグ格納部に上記第１の形態での演算器１における出力データたる条件フラグＴ０～Ｔ７を格納している。出力レジスタ８８は、１ビットのビット長を有する１０個のフラグ格納部Ｇ０～Ｇ９を有し、各フラグ格納部にそれぞれ所定のフラグ（Ｕ０～Ｕ７、ＵＰ、ＵＡ）が格納できるようにしてある。

【００４０】この演算器８０では、第１の入力レジスタ８２の各データ格納部Ｒ１０～Ｒ１７に格納された入力データＡ０～Ａ７と第２の入力レジスタ８４（即ちフラグレジスタ８）の各フラグ格納部Ｒ２０～Ｒ２７に格納された条件フラグＴ０～Ｔ７とをそれぞれ用いて演算部８６において同時に各データと条件フラグの組に共通の演算が行われ、その演算結果に対応したフラグＵ０～Ｕ７（０又は１）が出力レジスタ８８のフラグ格納部Ｇ０～Ｇ７に格納される。出力レジスタ８８のフラグ格納部Ｇ８には、フラグ格納部Ｇ０～Ｇ７に格納されている出力フラグＵ０～Ｕ７の論理和演算の結果に対応したフラグＵＰ（０又は１）が格納される。他方、出力レジスタのデータ格納部Ｇ９には、フラグ格納部Ｇ０～Ｇ７に格納されている出力フラグＵ０～Ｕ７の論理積演算の結果に対応したフラグＵＡ（０又は１）が格納される。

【００４１】図９において、出力レジスタ８８のフラグ格納部Ｇ０～Ｇ７に格納されるフラグＵ０～Ｕ７は、上記第１の実施の形態の演算器１の出力フラグと同様に、桁上がりを示すキャリーフラグである。

【００４２】図１０は図９に示す演算器の変形例を示す。第１の入力レジスタ９２は、１６ビットのビット長を有する４個のデータ格納部Ｒ１０～Ｒ１３で構成されており、各データ格納部に所定の演算データＡ０～Ａ３が格納できるようにしてある。第２の入力レジスタ９４、即ちフラグレジスタ１８は、１ビットのビット長を有する少なくとも４個のフラグ格納部Ｆ０～Ｆ３で構成されており、各フラグ格納部に上記第１の形態での演算器１０における出力データたる条件フラグＴ０～Ｔ３を

格納している。出力レジスタ９８は１ビットのビット長を有する６個のフラグ格納部Ｇ０～Ｇ５を有し、各フラグ格納部にそれぞれ所定のフラグ（Ｕ０～Ｕ３、ＵＰ、ＵＡ）が格納できるようにしてある。ここで、出力レジスタ９８のフラグ格納部Ｇ０～Ｇ３に格納されるフラグＵ０～Ｕ３は、上記図９の実施の形態の演算器８０のフラグＵ０～Ｕ７と同様に求められ格納される。出力レジスタ９８のフラグ格納部Ｇ４にはフラグＵＰが格納されるが、上記図９の実施の形態と概略同様で、フラグ格納部Ｇ０～Ｇ３に格納されている出力フラグＵ０～Ｕ３の論理和演算の結果に対応したものである。同様に、出力レジスタ９８のフラグ格納部Ｇ５にはフラグＵＡが格納され、フラグ格納部Ｇ０～Ｇ３に格納されている出力フラグＵ０～Ｕ３の論理積演算の結果に対応する。

【００４３】図１１も図９に示す演算器の変形例を示す。第１の入力レジスタ１０２は、３２ビットのビット長を有する２個のデータ格納部Ｒ１０～Ｒ１１で構成されており、各データ格納部に所定の演算データＡ０～Ａ１が格納できるようにしてある。第２の入力レジスタ１０４、即ちフラグレジスタ２８は、１ビットのビット長を有する少なくとも２個のフラグ格納部Ｆ０～Ｆ１で構成されており、各フラグ格納部に上記第１の形態での演算器２０における出力データたる条件フラグＴ０～Ｔ１を格納している。出力レジスタ１０８は１ビットのビット長を有する４個のフラグ格納部Ｇ０～Ｇ３を有し、各フラグ格納部にそれぞれ所定のフラグ（Ｕ０～Ｕ１、ＵＰ、ＵＡ）が格納できるようにしてある。ここで、出力レジスタ１０８のフラグ格納部Ｇ０～Ｇ１に格納されるフラグＵ０～Ｕ１は、上記図９の実施の形態の演算器８０のフラグＵ０～Ｕ７と同様に求められ格納される。出力レジスタ１０８のフラグ格納部Ｇ２にはフラグＵＰが格納されるが、上記図９の実施の形態と概略同様で、フラグ格納部Ｇ０～Ｇ１に格納されている出力フラグＵ０～Ｕ１の論理和演算の結果に対応したものである。同様に、出力レジスタ１０８のフラグ格納部Ｇ３にはフラグＵＡが格納され、フラグ格納部Ｇ０～Ｇ１に格納されている出力フラグＵ０～Ｕ１の論理積演算の結果に対応する。

【００４４】図１２もまた、図９に示す演算器の変形例を示す。第１の入力レジスタ１１２は、６４ビットのビット長を有する１個のデータ格納部Ｒ１０で構成されており、所定の演算データＡ０が格納できるようにしてある。第２の入力レジスタ１１４、即ちフラグレジスタ３８は、１ビットのビット長を有する少なくとも１個のフラグ格納部Ｆ０で構成されており、上記第１の形態での演算器３０における出力データたる条件フラグＴ０を格納している。出力レジスタ１１８は、１ビットのビット長を有する１個のフラグ格納部Ｇ０を有し、そのフラグ格納部には所定のフラグＵ０が格納できるようにしてある。ここで、出力レジスタ１１８のフラグ格納部Ｇ０に格納されるフラグＵ０は、上記図９の実施の形態の演算器８０のフラグＵ０～Ｕ７と同様に求められ格納される。

【００４５】このように構成された演算器によれば、先行して実行したＳＩＭＤ型演算の結果を後続するＳＩＭＤ型演算に演算単位で反映させるのが容易になる。

【００４６】図９の形態では、６４ビットのビット長の入力レジスタを備えた演算器を１個だけ用意し、その１個の演算器の内部で８つの演算を並列的に行うことができるので、同一ビット長の入力レジスタを有する演算器を８個も用意する必要がない。その結果、小さい回路規模を実現できる。図９の形態のみならず、図１０及び図１１においても同様である。

【００４７】本発明の第４の実施の形態の演算器を図１３に示す。演算器１２０は、入力レジスタ１２２と、演算部１２６と、出力レジスタ１２８とを有する。入力レジスタ１２２と、出力レジスタ１２８のビット長は６４ビットである。

【００４８】図１３の演算器１２０では、入力レジスタ１２２は、１６ビットのビット長を有する４個のデータ格納部Ｒ１０～Ｒ１３で構成されており、各データ格納部に所定の演算データＡ０～Ａ３が格納できるようにしてある。出力レジスタ１２８は、１６ビットのビット長を有する４個のデータ格納部Ｒ３０～Ｒ３３で構成されており、各データ格納部に所定の演算結果データＺ０～Ｚ３が格納できるようにしてある。

【００４９】この演算器１２０においては、入力レジスタ１２２の各データ格納部Ｒ１０～Ｒ１３に格納された入力データＡ０～Ａ３をそれぞれ用いて演算部１２６のにおいて同時に各データに共通の演算が行われるが、その際、上記第１の実施の形態での演算器１０におけるフラグレジスタ１８に格納される条件フラグＴ０～Ｔ３のおのおのが、演算部１２６での各演算に条件を与える。その演算結果Ｚ０～Ｚ３が出力レジスタ１２８のデータ格納部Ｒ３０～Ｒ３３に格納される。

【００５０】上記図１３の実施の形態の演算器について具体例を図１４に示す。図１４においては、入力レジスタ１３２に格納されているＡ０～Ａ３の４つのデータを条件フラグＴ０～Ｔ３の値により、符号変換する様子を示す。Ａ０は‘１２’であり、対応する条件フラグＴ０が‘１’であるため符号変換を行い、演算結果データＺ０は‘－１２’となる。Ａ１は‘－５６’であり、対応する条件フラグＴ１が‘０’であるため符号変換を行わず、演算結果データＺ１はそのまま‘－５６’となる。Ａ２及びＡ３についても同様の変換を行う。

【００５１】このように構成された演算器によれば、先行して実行したＳＩＭＤ型演算の結果によって、条件を満足（あるいは不満足）している演算データに対する処理を選択的に実行できる。

【００５２】図１３、図１４の形態では、６４ビットの

ビット長を有する入力レジスタを備えた演算器を１個だけ用意し、その１個の演算器の内部で４つの演算を並列的に行うことができるので、同一ビット長の入力レジスタを有する演算器を４個も用意する必要がない。その結果、小さい回路規模を実現できる。

【００５３】本発明の第５の実施の形態の演算器を図１５に示す。演算器１４０は、第１の入力レジスタ１４２と、第２の入力レジスタ１４４と、演算部１４６と、出力レジスタ１４８とを有する。本構成は、上記の第１の実施の形態の演算器１０と概略同様の構成である。但し、第２の入力レジスタ１４４がただ１個だけのデータ格納部Ｂ０を有することが、異なる点である。

【００５４】第１の入力レジスタ１４２は、８ビットのビット長を有する８個のデータ格納部Ｒ１０～Ｒ１７で構成されており、各データ格納部に所定の演算データＡ０～Ａ７を格納できるようにしてある。第２の入力レジスタ１４４は、８ビットのビット長を有する１個のデータ格納部Ｒ２０で構成されており、そこに所定の演算データＢ０を格納できるようにしてある。出力レジスタ１４８は、１ビットのビット長を有する１０個のフラグ格納部Ｆ０～Ｆ９を有し、各フラグ格納部にそれぞれ所定のフラグ（Ｔ０～Ｔ７、ＴＰ、ＴＡ）が格納できるようにしてある。

【００５５】この演算器１４０では、第１の入力レジスタ１４２の各データ格納部Ｒ１０～Ｒ１７に格納された入力データＡ０～Ａ７と第２の入力レジスタ１４４の１個のデータ格納部Ｒ２０に格納された入力データＢ０とをそれぞれ用いて演算部１４６において同時に各データの組に共通の演算が行われ、その演算結果に対応したフラグＴ０～Ｔ７（０又は１）が出力レジスタ１４８のフラグ格納部Ｆ０～Ｆ７に格納される。出力レジスタ１４８のフラグ格納部Ｆ８には、フラグ格納部Ｆ０～Ｆ７に格納されている出力フラグＴ０～Ｔ７の論理和演算の結果に対応したフラグＴＰ（０又は１）が格納される。他方、出力レジスタ１４８のデータ格納部Ｆ９には、フラグ格納部Ｆ０～Ｆ７に格納されている出力フラグＴ０～Ｔ７の論理積演算の結果に対応したフラグＴＡ（０又は１）が格納される。

【００５６】図１５において、出力レジスタ１４８のフラグ格納部Ｆ０～Ｆ７に格納されるフラグＴ０～Ｔ７は、上記第１の実施の形態の演算器１の出力フラグと同様に、桁上がりを示すキャリーフラグである。

【００５７】このように構成された演算器によれば、上記第１の実施の形態の演算器１０の場合で得られる効果のみならず、例えば同じデータを複数のデータに対して加算したい場合、予め同じデータＢ０をＢ１～Ｂ７に並列に並べる手間が省け、高速化に寄与するという、固有の利点がある。

【００５８】第３の実施の形態の演算器に関する上記説明においては、第１の実施の形態の演算器により予め出力されている出力レジスタを、第２の入力レジスタとして用いるとしているが、この第２の入力レジスタとして用いる手段は、第３の実施の形態の演算器により予め出力されている出力レジスタであってもよい。同様に、第２の実施の形態の演算器での第２の入力レジスタや、第４の実施の形態の演算器での演算に条件を与えるレジスタとして用いる手段も、第３の実施の形態の演算器により予め出力されている出力レジスタであってもよい。

【００５９】次に本発明の演算器で得られる結果を応用した処理のプログラム例を示す。ここで示す例は、コードのパターンマッチングを行うものである。表１に示すようなテーブルに基づいて多量のデータをコード変換により圧縮し、この圧縮データを蓄積あるいは通信する場合に利用される。あるデータを圧縮する場合、そのデータよりも符号量の小さいコードへの変換が行われる。逆に、圧縮されたデータは、圧縮されたコードから復号値を得て、即ち伸張をして利用することになる。

【００６０】

【表１】

| 符号値 | 復号値 |
|---|---|
| Ｃ０ | Ｖ０ |
| Ｃ１ | Ｖ１ |
| Ｃ２ | Ｖ２ |
| Ｃ３ | Ｖ３ |
| Ｃ４ | Ｖ４ |
| Ｃ５ | Ｖ５ |
| Ｃ６ | Ｖ６ |
| Ｃ７ | Ｖ７ |
| | |

【００６１】表１の符号化／複合化テーブルによる圧縮／伸張の手順を例示する。圧縮したいデータが‘Ｖ５’であれば、これをもとに表１から圧縮された符号Ｃ５を得る。逆に伸張する場合は、圧縮され、かつ復号値が未知のコードを表１のＣ０、Ｃ１、Ｃ２、・・・・と比較していき、一致したときの復号値をテーブルから得る。本例の場合、コードが‘Ｃ５’に一致したとき、そのときの復号値Ｖ５を得る。このような圧縮／伸張方法の代表的なものの一つとして、当業者には周知の手段である可変長（ハフマン）符号化／復号化があり、これらはＭＰＥＧの画像圧縮／伸張等に広く利用されている。以下に、表１の符号化／復号化テーブルを用い、圧縮データ

から復号値を得る伸張処理の例を示す。

【００６２】図１６は、当該伸張処理で使用するレジスタ群の構成を示す。レジスタＲ０に格納されているｘが圧縮コードであり、レジスタＲ１に格納するｙが求める復号値である。レジスタＲ２は、符号値テーブルが格納されているメモリ上の先頭（ベース）アドレスであり、レジスタＲ３は復号値テーブルが格納されているメモリ上の先頭（ベース）アドレスである。レジスタＲ４とレジスタＲ５はいずれもワーキングレジスタである。符号値テーブルと復号値テーブルのメモリ上の格納の様子を図１７に示す。当該伸張処理例では符号値及び復号値共に１６ビット（２バイト）データである。以上のような条件をもとに、復号値が未知である符号コードｘの復号値を得るためのフローチャートを図１８に示す。

【００６３】図１８のフローチャートに沿って復号値を得るためにアセンブリ言語で組まれたプログラムの例を図１９に示し、またその詳細な処理内容も図中に示す。

【００６４】図１９において、第２行から第５行が表１の符号化／復号化テーブル上の最初のコードＣ０と符号コードｘとの比較を示す。第２行にてレジスタＲ４にメモリ上の符号値テーブルの最初の符号値Ｃ０を格納する。第３行にてこのコードを格納するレジスタＲ４と、復号値が未知である符号コードｘを格納するレジスタＲ０との、比較演算を行い、その結果、一致すれば一致したことを示すフラグＴ０（条件フラグ）をＴ０＝‘１’として１ビット出力する。この比較の演算は、加算の例を示した演算器の第１の実施の形態で、特に図４において、加算演算と比較演算とを置き換えたものである。上記第３行にて符号が一致し、ビットＴ０に‘１’が格納された場合には、第４行にてＴ０の内容に従い分岐処理を行う。分岐先は第３０行からの復号値ロードシーケンスである。符合が一致せずＴ０に‘０’が格納された場合、第４行で分岐せず、以下第５行へと進む。

【００６５】第５行では、上記比較演算にて一致しなかったことをうけて復号値テーブルアドレスＲ３をインクリメントする。以後、概略、上記第２行から第５行のような比較演算のシーケンスを、比較対象コードを順に変えつつ繰り返すことになる。ここで、復号値が未知である符号コードｘがＣ５である場合、図１９の第２９行で符号Ｃ５との一致を検出し、第３０行の復号値ロードシーケンスに分岐する。以上の実施例では符号がＣ５のとき、復号処理を終了するのに２４命令を実行する必要があることがわかる。

【００６６】上記の例においては、レジスタＲ０とレジスタＲ４に１個ずつコードを格納して比較演算処理を行い、結果に対応するフラグＴ０を１ビット出力し、そのフラグを判断条件として分岐処理を行うが、２つの入力レジスタにそれぞれ複数のコードを格納し、ＳＩＭＤ方式で演算処理を行いフラグを複数（Ｔ０、Ｔ１、・・・Ｔｎ）ビット出力し、それぞれのフラグを判断条件として分岐処理を行うことも可能である。

【００６７】図２０ではさらに条件論理和フラグを利用した復号処理のプログラム例を示す。第２行から第５行が４つの符号コード（符号Ｃ０、Ｃ１、Ｃ２、Ｃ３）と、復号値が未知である符号コードｘとの、ＳＩＭＤ方式による比較演算部分である。第２行でこれら４つの符号コード（計６４ビット）をメモリからレジスタＲ４に格納し、第３行で復号値が未知である符号コードｘとの比較を行っている。この比較演算処理の演算器の様子を図２１に示す。

【００６８】図２１において、レジスタＲｓ１のデータ格納部Ａ０～Ａ３に符号コードＣ０～Ｃ３が格納され、レジスタＲｓ２のデータ格納部Ｂ０に復号値が未知である符号コードｘが格納される。ここでは、第５の実施の形態の演算器１４０が応用されている。各比較演算の結果が条件フラグＴ０～Ｔ３に設定され、Ｔ０～Ｔ３を元に条件論理和フラグＴＰと条件論理積フラグＴＡの内容が設定され、これらフラグはフラグレジスタの対応するフラグ格納部に格納される。

【００６９】図２０の第４行にて、４つの比較演算のうちで一致が発生したかどうかを条件論理和フラグＴＰにより判断し、一致があった場合は第１０行以降の復号値ロードシーケンスに分岐する。一致がなかった場合は、次の符号コードＣ４、Ｃ５、Ｃ６、Ｃ７との比較を行うため符号テーブルのベースアドレスＲ２をコード４つ分（計８バイト）インクリメントする。図１９の例で示したように復号値が未知であるコードｘがＣ５である場合、図２０の第９行でＣ４、Ｃ５、Ｃ６、Ｃ７のいずれかと一致したことが条件論理和フラグＴＰにより検出され、第１０行以降の復号値ロードシーケンスに分岐する。

【００７０】第１１行の‘ＴＳＣＨ　Ｒ５’命令は、フラグレジスタ上のフラグＴ３、Ｔ２、Ｔ１、Ｔ０を左（上位）側から検索し、レジスタ上の最初の‘１’の位置をＲ５に格納する。本実施例では復号値は２バイトデータであるため、第１１行で得た一致位置の値を第１２行で２倍し、第１３行にて直前に求めたアドレス増分Ｒ５と復号テーブルベースアドレスＲ３との加算値をアドレスとして、レジスタＲ１に復号値をロードし、復号処理は終わりとなる。この例で、未知のコードｘがＣ５の場合、実行される命令数は１０となり、前述の例の図１９の２４命令に比べ、半分以下に実行命令数が減少しており、処理が高速化されることがわかる。

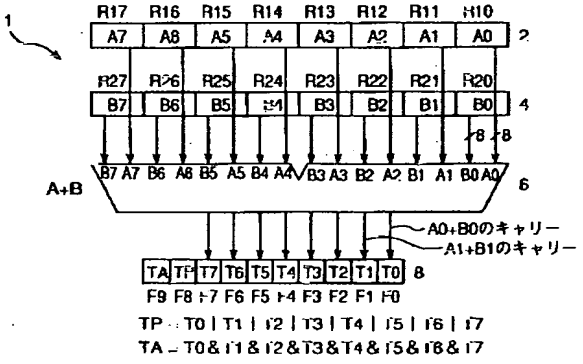
【００７１】前述したＭＰＥＧ復号処理では、このような復号化処理が大量に行われるため、本発明による高速化の効果はかなり大きいものになる。

【００７２】上記の例においては、条件論理和フラグを判断条件として分岐処理を行うが、プログラムのロジック次第では分岐処理の判断条件として、条件論理積フラグを用いて分岐処理を行うこともあり得る。

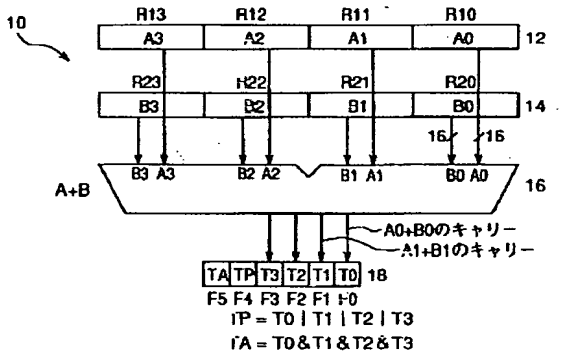
【００７３】
【発明の効果】以上の説明から明らかなように、本発明のＳＩＭＤ式の演算処理装置によれば、以下のような効果が存する。
【００７４】２つの入力手段間で対応する各データ格納部に格納される２つのデータを用いてそれぞれ同時に共通の演算をし、それぞれの演算結果に対応する条件フラグを出力手段の対応するフラグ格納部に格納する、本発明のＳＩＭＤ型演算器では、１ステップの命令実行であっても複数かつ共通の演算が同時に行われ、それら結果から複数の条件を生成できるので、単一の演算を繰り返して複数の条件を生成するよりも、費やす時間が少なくて済み処理の高速化につながる。また、条件を満足する（あるいは不満足な）演算がどれかを検索することが容易となる。さらに、条件フラグの論理和である条件論理和フラグを出力すれば、一度に演算した複数の演算データの全てが条件に不満足なのか、それとも少なくともひとつ以上は条件に満足なのかが、この条件論理和フラグを確認するだけで判明する。同様に条件論理積フラグを出力すれば、一度に演算した複数の演算データの全てが条件に満足なのか、それとも少なくともひとつ以上は条件に不満足なのかが、この条件論理積フラグを確認するだけで判明する。
【００７５】上記のＳＩＭＤ演算器において、第２の入力手段が少なくとも第１の入力手段のデータ格納部のビット長以上の長さであり、かつ第１の入力手段のデータ格納部と長さが等しい１個のデータ格納部を有し、第１の入力手段の各データ格納部に格納されるデータと第２の入力手段の１個のデータ格納部に格納されるデータとを用いて同時に各データの組に共通の演算を行うものである場合、特に複数データに対して一定の数値を加算するようなときに、処理速度、命令の指定の容易さ、の点で有利である。
【００７６】入力手段のうちの１つが上記のＳＩＭＤ演算器の出力手段、即ちフラグレジスタである、本発明のＳＩＭＤ演算器では、先行して実行した演算の結果を後続する演算に反映することができる。
【００７７】フラグレジスタに格納される条件フラグを、入力手段の各データ格納部に格納されるデータに対応付けて、条件フラグの内容により演算に条件を与える、本発明のＳＩＭＤ演算器では、先行して実行したＳＩＭＤ演算の結果によって、条件を満足（あるいは不満足）しているデータだけに対する処理を選択的に実行することができる。条件フラグに対応する演算データ単位の実行内容の変更ができなければ、条件を満足（あるいは不満足）しているデータだけを抽出して処理するか、条件を満足（あるいは不満足）しないデータに対し後続の処理の影響が与えられないような工夫が必要であり、処理速度、処理の容易さの点で、不利である。
【００７８】本発明のＳＩＭＤ演算器にて生成される条件フラグを分岐処理の判断条件とする条件分岐処理機能を備えるＣＰＵでは、演算は一度に実行しても結果の条件フラグによりその後個別の処理を与えることができる。さらに、条件論理和フラグまたは条件論理積フラグを分岐処理の判断条件とする条件分岐処理機能により、それぞれのフラグの内容による個別の処理を設定することが可能となる。
【００７９】上記条件フラグを格納した出力手段上で最上位（あるいは最下位）に位置する“１”（あるいは“０”）を格納したフラグ格納部の位置を数値化する機能を備えるＣＰＵでは、条件を満足（あるいは不満足）した演算が、どの演算であったのか容易に判明しうる。この機能がない場合は、条件フラグの個々について“１”（あるいは“０”）の有無を検査し、初めて見つかった“１”（あるいは“０”）の条件フラグが所望のデータの位置であるとするプログラムにより位置の数値化を行わねばならず、処理速度、容易さの点で不利である。
【図面の簡単な説明】
【図１】　ＳＩＭＤ演算器による条件フラグ生成の説明図（１）。
【図２】　ＳＩＭＤ演算器による条件フラグ生成の説明図（２）。
【図３】　ＳＩＭＤ演算器による条件フラグ生成の説明図（３）。
【図４】　ＳＩＭＤ演算器による条件フラグ生成の説明図（４）。
【図５】　条件フラグを用いたＳＩＭＤ演算器による演算の説明図（１）。
【図６】　条件フラグを用いたＳＩＭＤ演算器による演算の説明図（２）。
【図７】　条件フラグを用いたＳＩＭＤ演算器による演算の説明図（３）。
【図８】　条件フラグを用いたＳＩＭＤ演算器による演算の説明図（４）。
【図９】　条件フラグを用いたＳＩＭＤ演算器による演算の説明図（５）。
【図１０】　条件フラグを用いたＳＩＭＤ演算器による演算の説明図（６）。
【図１１】　条件フラグを用いたＳＩＭＤ演算器による演算の説明図（７）。
【図１２】　条件フラグを用いたＳＩＭＤ演算器による演算の説明図（８）。
【図１３】　条件フラグビットによるＳＩＭＤ演算制御の説明図。
【図１４】　条件フラグビットによるＳＩＭＤ演算制御の具体的説明図。
【図１５】　ブロードキャスト方式を用いた本発明の条件フラグ生成の説明図。
【図１６】　伸張処理で使用するレジスタの説明図。

【図17】　符号値と復号値のメモリ上の格納の説明図。
【図18】　符号コードxの復号値を得るためのフローチャート。
【図19】　符号コードxの復号値を得るためのプログラム。
【図20】　条件論理和フラグを利用した符号コードxの復号値を得るためのプログラム。
【図21】　図20のプログラムでのブロードキャスト方式を用いた比較演算の説明図。
【符号の説明】
1、10，20，30・・・SIMD型演算器、2、12、22、32・・・第1の入力レジスタ、4、14、24、34・・・第2の入力レジスタ、6、16、26、36・・・演算部、8、18、28、38・・・出力レジスタ、40、50，60，70・・・SIMD型演算器、42、52、62、72・・・第1の入力レジスタ、44、54、64、74・・・第2の入力レジスタ、46、56、66、76・・・演算部、48、58、68、78・・・出力レジスタ、80、90，100，110・・・SIMD型演算器、82、92、102、112・・・第1の入力レジスタ、84、94、104、114・・・第2の入力レジスタ、86、96、106、116・・・演算部、88、98、108、118・・・出力レジスタ、120、130・・・SIMD型演算器、122、132・・・入力レジスタ、126、136・・・演算部、128、138・・・出力レジスタ、140・・・SIMD型演算器、142・・・第1の入力レジスタ、144・・・第2の入力レジスタ、146・・・演算部、148・・・出力レジスタ、Rs1・・・レジスタ、Rs2・・・レジスタ

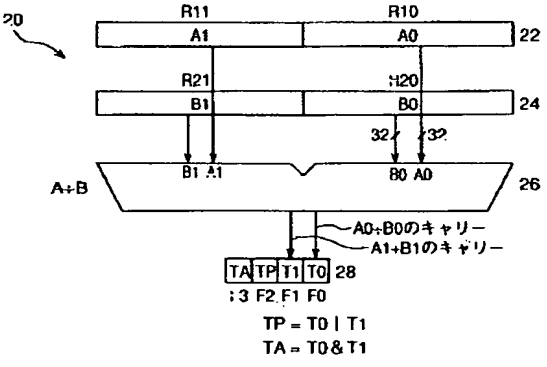
【図1】



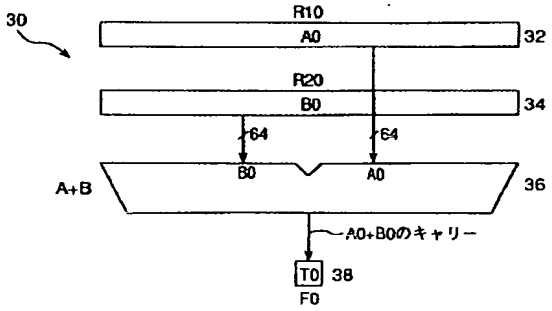

【図2】



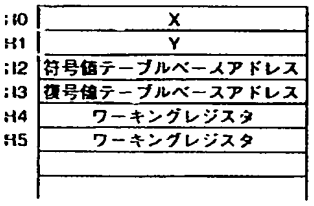

【図3】



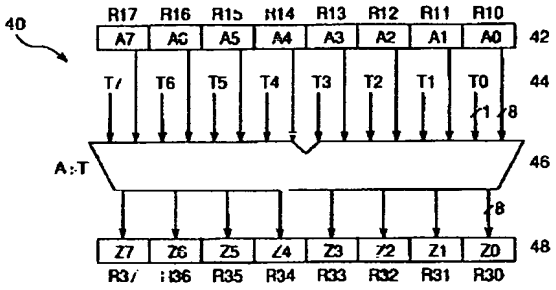

【図4】



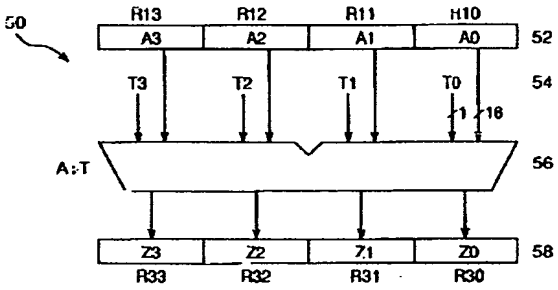

【図16】



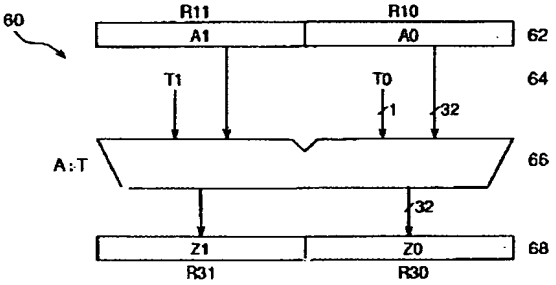
【図5】



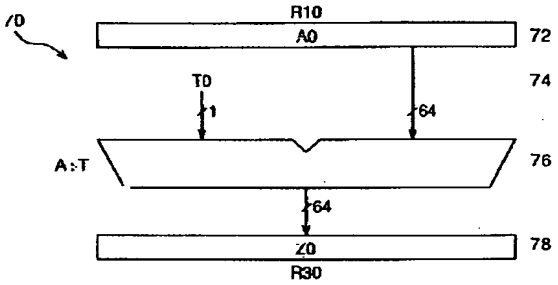

【図6】



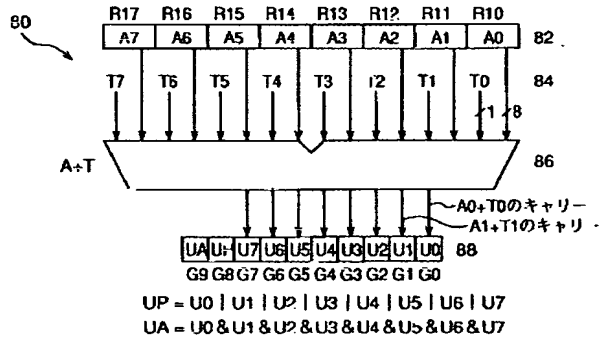

【図7】



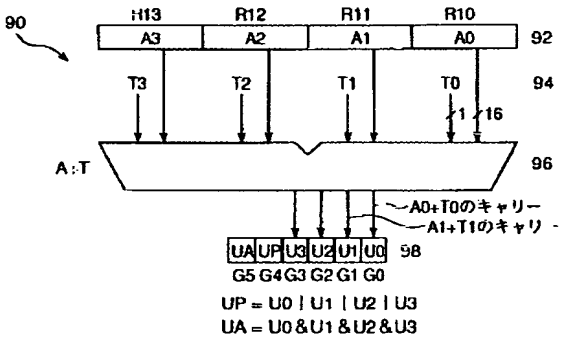

【図8】



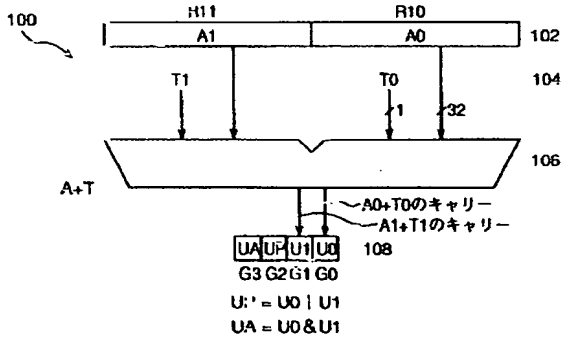

【図9】



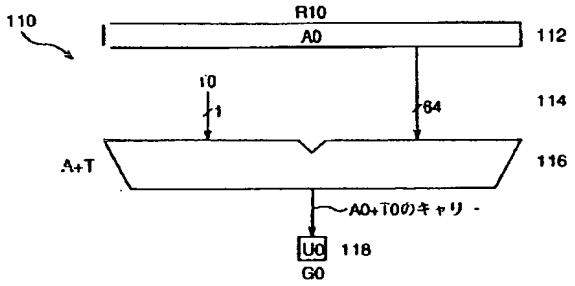

【図10】



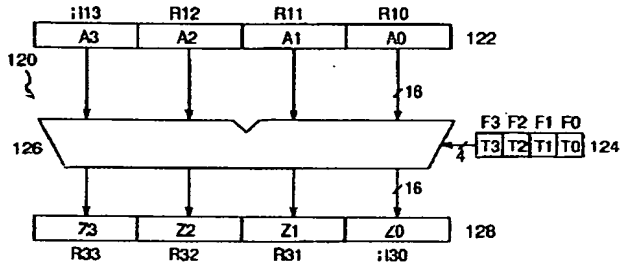

【図11】



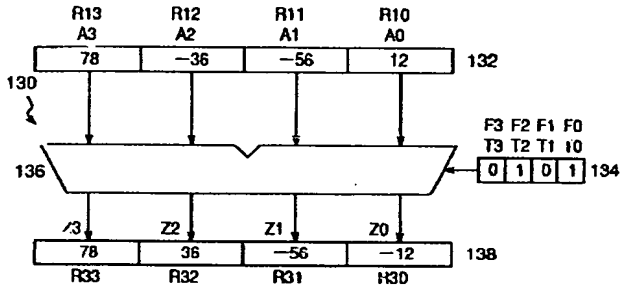

【図12】



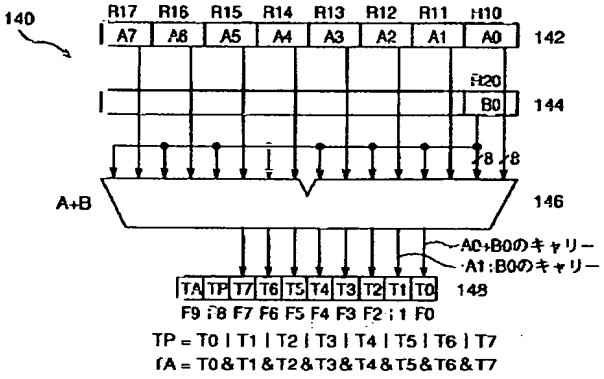
【図13】



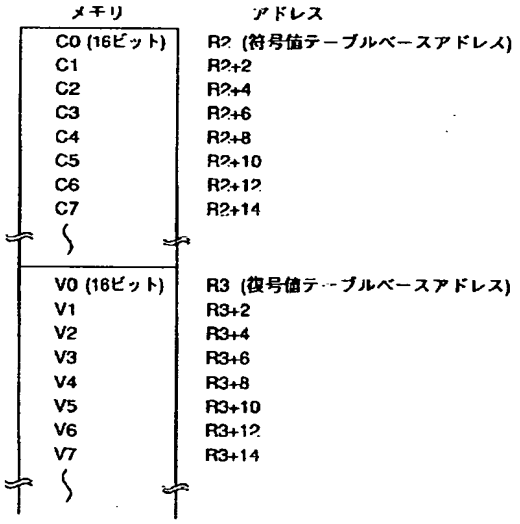

【図14】



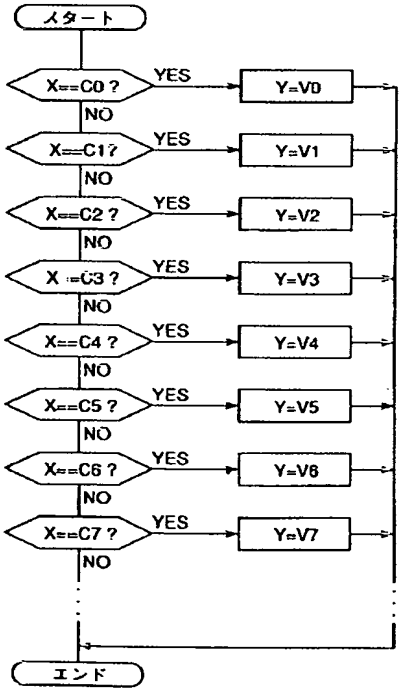

【図15】




【図17】




【図18】



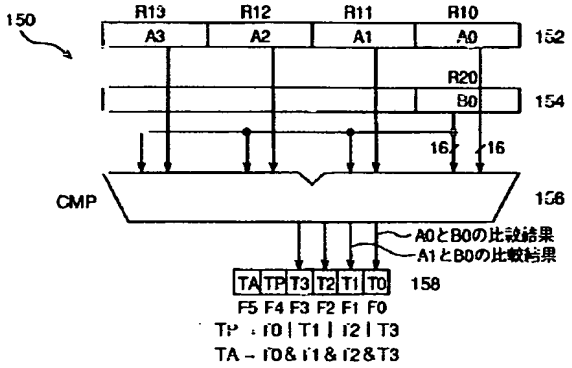

【図20】

```
行番号
 1  CMP0123:
 2          LD. d    R4, 0(R2)   ; R4にメモリ上の符号C0,C1,C2,C3をロード
 3          CMP. hc  R0, R4      ; R0(x)とR4(C0,C1,C2,C3)を比較、結果はT0～T3へ反映
 4          BRE/T0   1, D: TECT ; TP(=T0,T1,T2,T3)が'1'のときDETECTへジャンプ
 5          ADD      R3, R3, 8  ; 復号値ベ－スアドレスR3を更新 (R3=R3+8)
 6  CMP4567:
 7          LD. d    R4, 8(R2)  ; R4にメモリ上の符号C4,C5,C6,C7をロード
 8          CMP. hc  R0, R4      ; R0(x)とR4(C4,C5,C6,C7)を比較、結果はT0～T3へ反映
 9          BRE/T0   1, D: TECT ; TP(=T0,T1,T2,T3)が'1'のときDETECTへジャンプ
                     :
10  D: TECT:
11          TSCH     R5         ; フラグレジスタ(T3,T2,T1,T0)を左から'1'の位置を検出
12          SHL      R5, R5, 4  ; R5を4ビット左(上位)シフト(R5を2倍)
13          LD. h    R1, (R5,R3); 復号値をメモリ上(アドレス=R5+R3)から得る。
14          NOP                 ; 復号処理終わり
```

【図19】

```
行番号
 1  CMP0:
 2          LD. h   R4, 0(R2)    ; R4にメモリ上の最初の符号C0をロード
 3          CMP     R0, R4       ; R0(x)とR4(C0)を比較、一致時T0=1、不一致時T0=0
 4          BNE/T0  1, DETECT    ; T0が'1'のときDETECTへジャンプ
 5          ADD     R3, R3, 2    ; 復号値ベースアドレス更新 (R3=R3+2)
 6  CMP1:
 7          LD. h   R4, 2(R2)    ; R4にメモリ上の符号C1をロード
 8          CMP     R0, R4       ; R0(x)とR4(C1)を比較、一致時T0=1、不一致時T0=0
 9          BNE/T0  1, DETECT    ; T0が'1'のときDETECTへジャンプ
10          ADD     R3, R3, 2    ; 復号値テーブルアドレス更新 (R3=R3+2)
11  CMP2:
12          LD. h   R4, 4(R2)    ; R4にメモリ上の符号C2をロード
13          CMP     R0, R4       ; R0(x)とR4(C2)を比較、一致時T0=1、不一致時T0=0
14          BNE/T0  1, DETECT    ; T0が'1'のときDETECTへジャンプ
15          ADD     R3, R3, 2    ; 復号値テーブルアドレス更新 (R3=R3+2)
16  CMP3:
17          LD. h   R4, 6(R2)    ; R4にメモリ上の符号C3をロード
18          CMP     R0, R4       ; R0(x)とR4(C3)を比較、一致時T0=1、不一致時T0=0
19          BNE/T0  1, DETECT    ; T0が'1'のときDETECTへジャンプ
20          ADD     R3, R3, 2    ; 復号値テーブルアドレス更新 (R3=R3+2)
21  CMP4:
22          LD. h   R4, 8(R2)    ; R4にメモリ上の符号C4をロード
23          CMP     R0, R4       ; R0(x)とR4(C4)を比較、一致時T0=1、不一致時T0=0
24          BNE/T0  1, DETECT    ; T0が'1'のときDETECTへジャンプ
25          ADD     R3, R3, 2    ; 復号値テーブルアドレス更新 (R3=R3+2)
26  CMP5:
27          LD. h   R4, 10(R2)   ; R4にメモリ上の符号C5をロード
28          CMP     R0, R4       ; R0(x)とR4(C5)を比較、一致時T0=1、不一致時T0=0
29          BNE/T0  1, DETECT    ; T0が'1'のときDETECTへジャンプ
                    :
30  DETECT:
31          LD. h   R4, 0(R3)    ; 一致したコードに対する復号値をR1(y)にロード
32          NOP                  ; 復号処理終わり
```

【図21】




フロントページの続き

(72)発明者　山浦　慎一
東京都大田区中馬込１丁目３番６号　株式会社リコー内

(72)発明者　門脇　幸男
東京都大田区中馬込１丁目３番６号　株式会社リコー内